AFRL-RI-RS-TR-2011-203

# DEFINING & DEMONSTRATING CAPABILITIES FOR EXPERIENCE-BASED NARRATIVE MEMORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY (MIT)

*JULY 2011*

FINALTECHNICAL REPORT



**STINFO COPY**

## AIR FORCE RESEARCH LABORATORY
## INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

# NOTICE AND SIGNATURE PAGE



AFRL-RI-RS-TR-2011-203 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
NANCY A. ROBERTS
Work Unit Manager

/s/
ANTHONY W. WILLIS, Col, USAF, Chief
Information & Intelligence Exploitation Division
Information Directorate

# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| JUL 2011 | Final Technical Report | JAN 2010 – JAN 2011 |

**4. TITLE AND SUBTITLE**
DEFINING & DEMONSTRATING CAPABILITIES FOR EXPERIENCE-BASED NARRATIVE MEMORY

**5a. CONTRACT NUMBER** FA8750-10-1-0076
**5b. GRANT NUMBER** N/A
**5c. PROGRAM ELEMENT NUMBER** 62304E

**6. AUTHOR(S)**
Mark A. Finlayson
Patrick H. Winston

**5d. PROJECT NUMBER** BC03
**5e. TASK NUMBER** 00
**5f. WORK UNIT NUMBER** 01

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Massachusetts Institute of Technology
32 Vassar Street
Room 32-251
Cambridge, MA 02138

**8. PERFORMING ORGANIZATION REPORT NUMBER**
N/A

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Air Force Research Laboratory/RIEH
525 Brooks Road
Rome NY 13441-4505

**10. SPONSOR/MONITOR'S ACRONYM(S)**
AFRL/RI

**11. SPONSORING/MONITORING AGENCY REPORT NUMBER**
AFRL-RI-RS-TR-2011-203

**12. DISTRIBUTION AVAILABILITY STATEMENT**
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2011-3985
Date Cleared: 21 JUL 2011

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
Researchers at MIT have constructed a novel proof-of-concept demonstration of memory-driven narrative structuring of information. Their demo system is a confederation of three systems they are developing: The Story Workbench semi-automatic annotation tool, the Genesis Commonsense Reasoning and Story Understanding system, and the Analogical Story Merging (ASM) algorithm. They first constructed linkages that allowed stories to be passed from the Story Workbench, through the Commonsense Reasoner, to the ASM system, so that higher-level plot patterns could be discovered automatically and returned to the Genesis system for discovery in an evolving story. They performed an experiment in which they constructed six stories, three of which illustrated the high-level plot pattern of revenge, the other three illustrating the high-level plot pattern of pyrrhic victory. They held on story of each type out for testing, and, using the confederated system, extracted both plot patterns from the remaining four stories. Using the patterns, they then were able to find the correct patterns in the held-out test stories. Also, in preparation for further experiments in this vein, they annotated a corpus of 16 Russian folktales in 16 different representations, approximately 20,000 words, which is the largest, most deeply-annotated narrative corpus assembled to date.

**15. SUBJECT TERMS**
memory-driven narrative structuring, pattern discovery, commonsense reasoning, semi-automatic annotation, plot detection, natural language processing, machine learning

| 16. SECURITY CLASSIFICATION OF: a. REPORT | b. ABSTRACT | c. THIS PAGE | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| U | U | U | UU | 21 | NANCY A. ROBERTS |
| | | | | | 19b. TELEPHONE NUMBER (*Include area code*) N/A |

# Table of Contents

## List of Figures

# 1. Introduction

The purpose of this project was to bring together three technologies in development at MIT: the Story Workbench, a semi-automatic annotation tool, a new technology for discovering patterns in sets of narratives, called Analogical Story Merging, and MIT's in-house multi-representational story understanding Genesis system. The marriage of these three technologies resulted in a novel proof-of-concept demonstration of a technique for memory-driven narrative structuring of information.

The agglomerated prototype system was a pipeline, the first element of which was the Story Workbench, which allows natural, college-level English to be translated semi-automatically into formal representations. These formal representations were then fed into the Genesis commonsense reasoning system, which inserts missing and elided information in the story. For example, given a brief synopsis of Shakespeare's Macbeth plot, the Genesis system can fill in the results of certain actions, such as that "if Ducan kills Macbeth, Macbeth is dead."

This information was then fed into the Analogical Story Merging (ASM) system, the third system we have been developing, which discovers common plot patterns using a novel modification of Bayesian Model Merging for extracting patterns from observed examples. For example, given a collection of five summaries of Shakespeare plays, the Analogical Story Merging System notes the detailed similarities of Macbeth and Hamlet, how Julius Caesar shares some structure with Macbeth and Hamlet but nowhere near as much, and how the Taming of the Shrew, a comedy, is different from the four dramas.

Finally, the plot patterns discovered by ASM were returned to the Genesis system, which Genesis searched the elaboration graph for patterns familiar to human readers, finding, for example, revenge and mistake patterns.

The search for such plot patterns is inspired, in part, by the work of Wendy Lehnert on what she called plot units and by the observation that description at the plot-unit level facilitates the recognition of precedents that usefully parallel new situations. An early guess at an applicable precedent enables focused information gathering to confirm or disconfirm the relevance of that precedent. A confirmed precedent enables prediction and intervention: for example, if you are playing the part of Macbeth in an unfolding Macbeth-like situation, it makes sense for you to determine if there is a potential revenge-seeking person; if so, you should seek out other precedents that show how to mollify or contain the revenge seeker.

Different domains, contexts and cultures have their own sets of plot patterns that vary in ways both significant and subtle. Accordingly, the key to success of the Genesis system in new domains is the ability to discover plot patterns given a narrative context. This project connected the Story Workbench, Genesis and Analogical Story Merging systems so as to enable an end-to-end demonstration that it is feasible to organize and filter an information stream according to stories drawn from a culturally-determined context.

Once the systems were connected, we ran a simple experiment. First, the Story Workbench was used to encode six stories for their formal meanings: three stories illustrating a *revenge*, and three illustrating a *pyrrhic victory*. These stories were passed to the Genesis System's commonsense inference module, and the resulting elaborated graphs for four of the stories (two of each) were fed into Analogical Story Merging. ASM extracted, automatically, the two plot-unit patterns, *revenge*, and *pyhrric victory*. These patterns were then fed back into the Genesis system so these two patterns could be detected in the two held-out test stories.

# 2. Detailed Technical Approach

The prototype system was constructed out of three parts, the Story Workbench, the Genesis System, and Analogical Story Merging.

## *The Story Workbench*

The Story Workbench is a tool for semi-automatically encoding computer representations of meaning, and allows a three-fold increase in speed over comparable annotation projects, and a four-fold reduction in costs, while still maintaining high quality annotations. Before the development of the Story Workbench, there were just two options for translating natural language into computer representations. The first was manually, using human annotators to generate the structures either by hand, or inside a specialized computer editor. This is slow, expensive, and error-prone. Alternatively, one could perform the analysis automatically – this is fast, but extremely inaccurate, and there are numerous representations that cannot be currently done this way.

The Story Workbench employs a *semi-automatic* annotation strategy. The tool, a screenshot of which is shown in Figure 1, does as much automatic processing as it can, using extant NLP technologies. Where it leaves off, a user-friendly user interface allows non-experts, with a minimal amount of training, to correct the analyses and make additions. The tool has numerous built-in rules that check the annotator's work, and also facilitates double-annotation, by providing ways of automatically merging and comparing the annotations of two different annotators.

This allows a significant increase in annotation speed, while still retaining the same accuracy as in manual annotation. For example, a good comparison annotation project is Project Halo, in which manual annotation by subject matter experts was estimated to cost at least $2,000/page (Angele 2003), with a rate for deep annotation of approximately 500 words/week. With the Story Workbench, we have been able to achieve a rate of 1500 words/week, a three-fold improvement, and also were able to use part-time, non-technical annotators, further reducing costs to $500/page, an overall four-fold improvement.

The Story Workbench allows for the annotation of 16 different layers of meaning, as follows:

1. Tokens - location of each word token
2. Multi-word Expressions - words that are made up of multiple tokens
3. Sentences - location of each sentence
4. Part of Speech Tags - a Penn Treebank tag for each word token and multi-word expression
5. Lemmas - a stem or root form for each word or multi-word expression not already lemmatized
6. Word Senses - a Wordnet sense for each token or multi-word expression
7. Referring Expressions - locations of all expressions that refer to something
8. Semantic Roles - predicate features and arguments, as defined in PropBank
9. Time Expressions - defined by TimeML (Pustejovsky 2003)

10. Events - location, features, and type of event mentions, as defined by TimeML
11. Referent Attributes - properties (unchanging attributes) of referents referred to in the text
12. Co-reference Relationships - which referring expressions refer to the same referent (co-refer)
13. Temporal Relationships - temporal relationships, as defined by TimeML
14. Referent Relationships - non-temporal relationships
15. Mental State - mental state valencies as described by Lehnert (Lehnert 1981)
16. Proppian Functions – locations of Propp's analyses of function

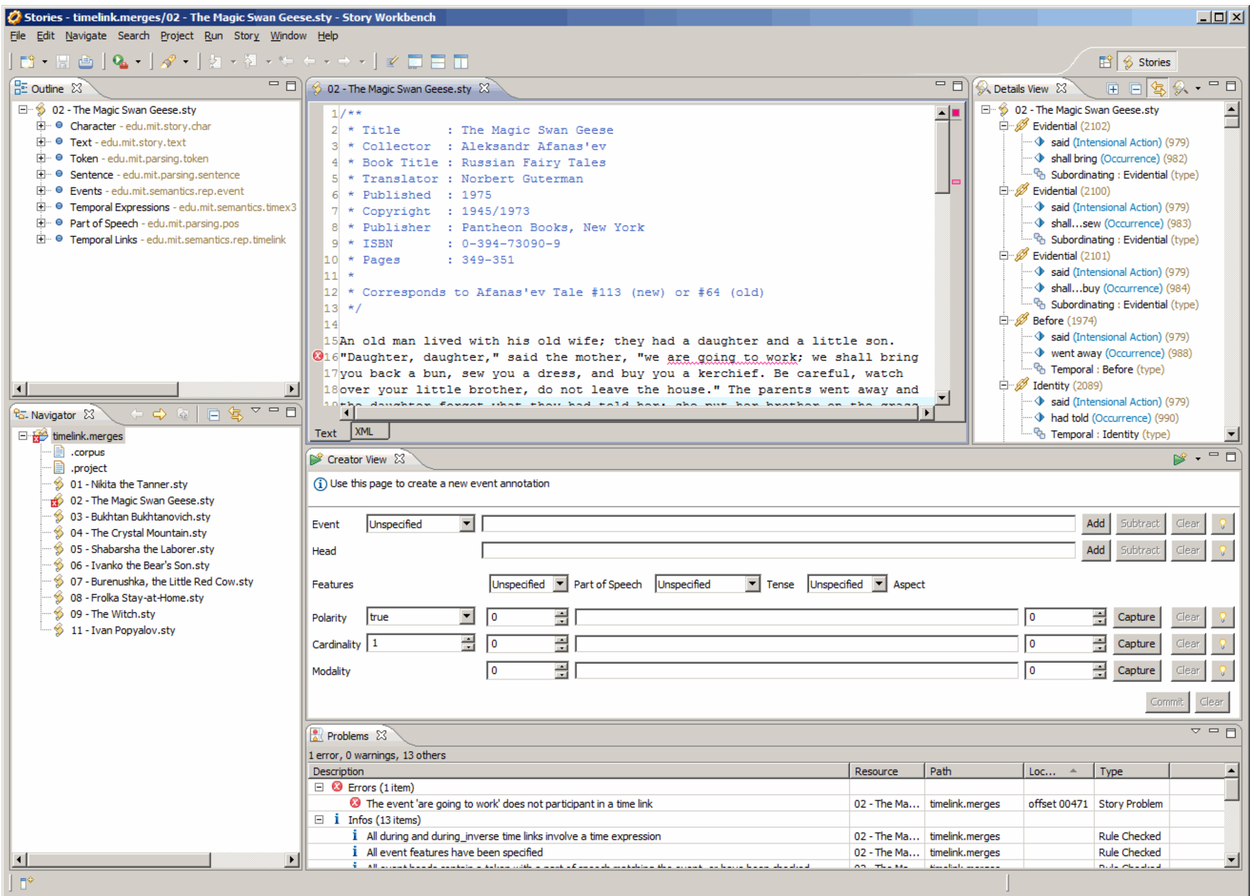


**Figure 1:** A screenshot of the Story Workbench. From the middle-top panel clockwise there is (1) the editor showing the current text being annotated; (2) the details view, showing the specific annotations in the representation currently being edited, which in this screenshot is the TimeML event representation; (3) the creator view, which allows fully manual creation and editing of annotations for representation currently being edited; (4) the problems view, which shows errors and warnings about the text being annotated; (5) the Navigator, which shows all available projects and files; and (6) the outline view, which shows all representations in for the text being edited.

## The Genesis System

The Genesis system is a confederated set of story understanding modules that encompasses tasks as diverse sentence parsing, co-reference resolution, event understanding, and commonsense inference.

The Genesis system is multi-representational, meaning that it represents its input in nearly two dozen frame-like representations. These include representations for threads (an approach to classification from Greenblatt and Vaina, 1979), trajectory (inspired by Jackendoff 1983), transition (inspired by Borchardt 1994) , transfer, location, time, cause, and coercion. There are many representations, in part, because there are many kinds of events to be described.

English descriptions instantiate these representations when we talk of physical-world events (the bird flew to a tree) as well as when we talk of abstract-world events (the country moved toward democracy). The particular representations we use were gathered, in part, from work by linguists and researchers in Artificial Intelligence. Others came from our own data-driven need to reflect the meanings encountered in the stories we use to drive our work.

Genesis work is representation-centric because we need representations to capture the constraints and regularities out of which we can build models, which in turn make it possible to understand, explain, predict, and control. Also, the bias toward multiple representations is inspired, in part, by Marvin Minsky's often articulated idea that if you have only one way of looking at a problem, you have no recourse if you get stuck.

The path from sentences to instantiated representations goes through the Start Parser, developed over a 25-year period by Boris Katz and his students (Katz et al 2002). We have used other, statistically trained parsers, but Start has two compelling advantages: Start blunders less and Start produces a semantic net, rather than a parse tree, making it much easier to instantiate our frame-like representations. We also exploit WordNet, using it as a source of classification information. Of course, we could get by without WordNet by supplying classification information in English (a Bouvier is a kind of dog) or by discovering it. Using WordNet is a temporary, time-saving shortcut.

## Analogical Story Merging

The final piece of the prototype system is a new computational technique for extracting higher-level patterns from natural language semantics called *Analogical Story Merging* (ASM). ASM is based on the machine learning technique of Bayesian Model Merging (Stolke & Omohundro 1994). Consider a toy example, where we wish to extract the similarities between two short stories:

(1) The boy and the girl were playing. He chased her, but she ran away. She thought he was gross.
(2) The man stalked the woman and scared her. She fled town. She decided he was crazy.

These two stories, dissimilar in specifics, are similar at higher level of abstraction: there is a pursuit, followed by a retreat and a judgment. To abstract away from the texts themselves to get at these higher-level patterns, we first must express the surface semantics of the texts, relatively fully, for the computer. This is shown schematically at the top of Figure 2, marked **D**, where the two stories have been represented as structured pieces of data marking each event in the stories, the agents and patients of each event, and the identities of the predicates involved.

The algorithm begins by constructing an initial model, marked $\boldsymbol{M_0}$ in Figure 2, which explicitly encodes each story as one possible output. This initial model represents the evidence that we have observed, and from which we want to extract patterns and in it, each piece of evidence (each story) is included in the model as a single linear branch. The model is much like a Finite State Machine or Markov Model, in that you can "generate" output from it by beginning at the start node, marked ***S***, and proceed along transitions to the next state of the model, choosing between multiple outgoing transitions according to their labeled probabilities.

To extract patterns, ASM then searches the space of *state merges*, where two states are merged into one. To accomplish merging, we define both a merge operation over states, and a *prior* probability function to be used when calculating, via Bayes' rule, the posterior probability of the model given the data. The merge operation takes two states and replaces them by a single state, where the merged state inherits the weighted sum of the transitions and emissions of its parents. Because each state in the initial model represents an event in the story, each merged state represents a set of all the events of its parent states.

The prior is defined such that smaller models are attributed greater probability than larger models, and models that contain merged states representing sets of similar events are given higher probability than otherwise. In ASM the primary calculation of similarity is done via an analogical mapping algorithm, an augmented version of the Structure Mapping Engine (Falkenhainer *et al.* 1989). This mapping algorithm assesses the similarity between two events, taking into account aspects of those events such as their structure (do the number of arguments match?), their classification (is it a *run* or a *love*?), the identities of other events to which the events in question are connected causally or temporally, and the consistency of role assignments (is character **A** in story 1 consistently mapped to character **B** in story 2?). Running the algorithm, it finds a path to the best model, i.e., the one that maximizes the posterior probability (the probability of the model given the data). Such a sequence of merges is shown in Figure 2.

As can be seen, the highly merged nodes represent exactly the higher-level structures we sought to extract, namely, that there is a pursuing event that leads to a retreat and judgment combination.

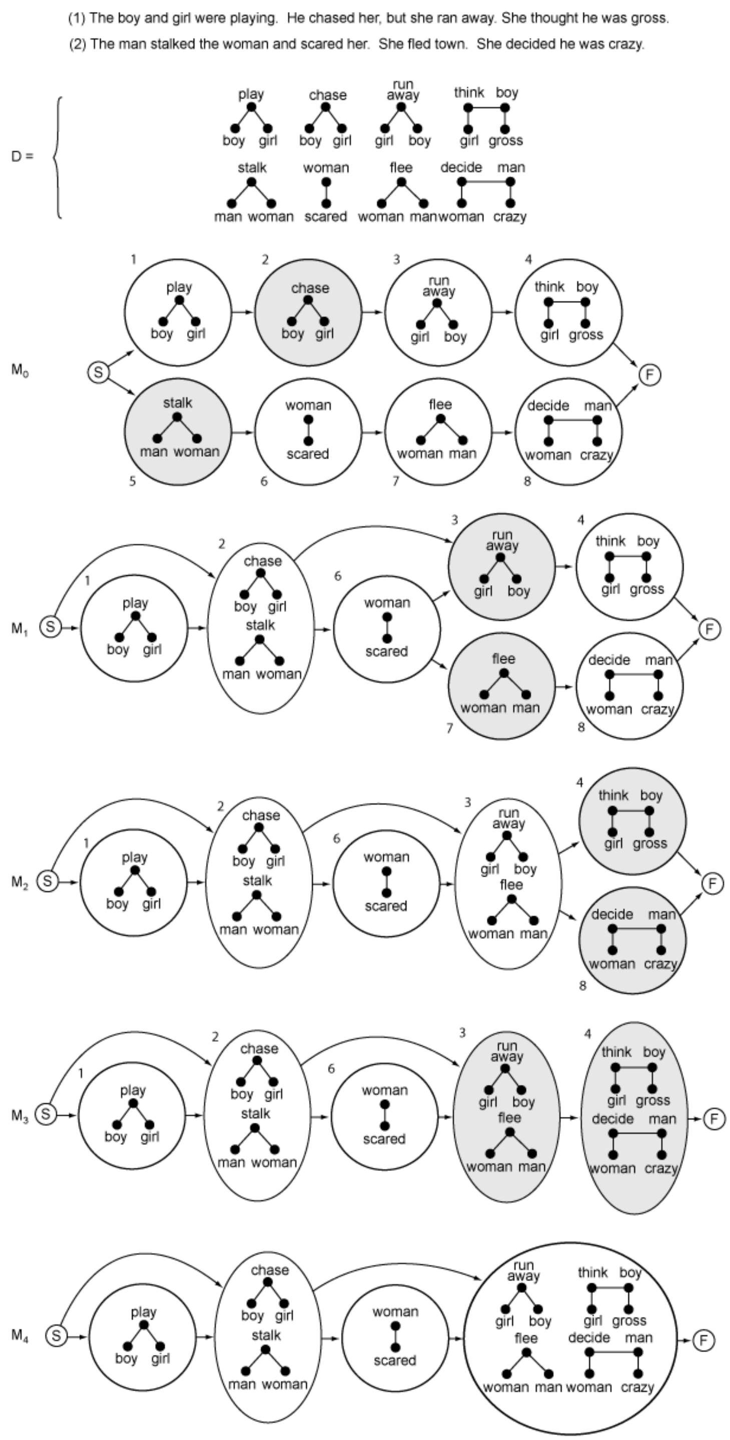


**Figure 2:** Analogical Story Merging in action. The two stories being merged are written at the top, in (1) and (2). The Story Workbench annotation step produces data structures representing the surface meaning of the story, marked here as ***D***. Each event in each story is then encapsulated in a single state, labeled 1 through 8, in the initial model $\boldsymbol{M_0}$. ASM searches the space of state merges to find a path to the most probable model, here labeled $\boldsymbol{M_4}$. From one model to the next, the two states that shaded in the first model are merged together in the second.

## 3. Experiment

In our experiment, the three systems described were chained together as shown in Figure 3.

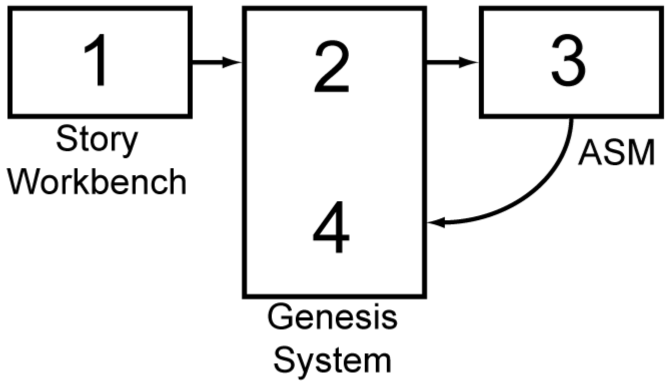


**Figure 3:** Flow chart of the prototype system. Stimuli used for the experiments were first processed with the Story Workbench (1), and then passed to the Genesis Commonsense Reasoner (2) for elaboration. The elaboration graphs so produced were then fed to Analogical Story Merging (3) and the plot patterns were extracted. Finally, the plot patterns were returned to the Genesis Plot Pattern detector (4) for identification in new stimuli.

The experiment demonstrated the successful marriage of the Genesis and Analogical Story Merging systems. First we constructed six stories: three stories illustrating a *revenge*, and three stories illustrating a *pyrrhic victory*. The stories are listed in Appendix A.

Once the six stories were annotated by the Story Workbench system, the formal representations so produced were fed into the Genesis Commonsense Reasoning system. This module of the Genesis system fills in gaps and adds common knowledge to the representation of the story, producing what we call the *elaboration graph*, shown in Figure 4. The commonsense knowledge used by the system was quite circumscribed: it is listed in full in Appendix B.

The passing of annotation from the Story Workbench to the Genesis system was the most difficult part of the infrastructure development to overcome. Both the Story Workbench and the Genesis system have separate suites of representations, with some overlap, but not a fully one-to-one mapping. This means there are some representations explicitly encoded on one side that are not encoded on the other. All information required for commonsense reasoning in Genesis, however, is present in the Story Workbench annotations; therefore it was a matter of writing a set of rules to transform the Story Workbench annotations into the Genesis representations.

The elaboration graph for two revenge stories and two pyrrhic victory stories was then fed into the Analogical Story Merging system, which processed them to produce two plot patterns representing *revenge* and *pyrrhic victory*. These two patterns were then fed back to the Genesis detection apparatus, shown in Figure 5. The third revenge and pyrrhic victory stories, unused so far, were then tested with the Genesis detection system, and the appropriate plot pattern was successfully found in each. This showed the prototype was successful.

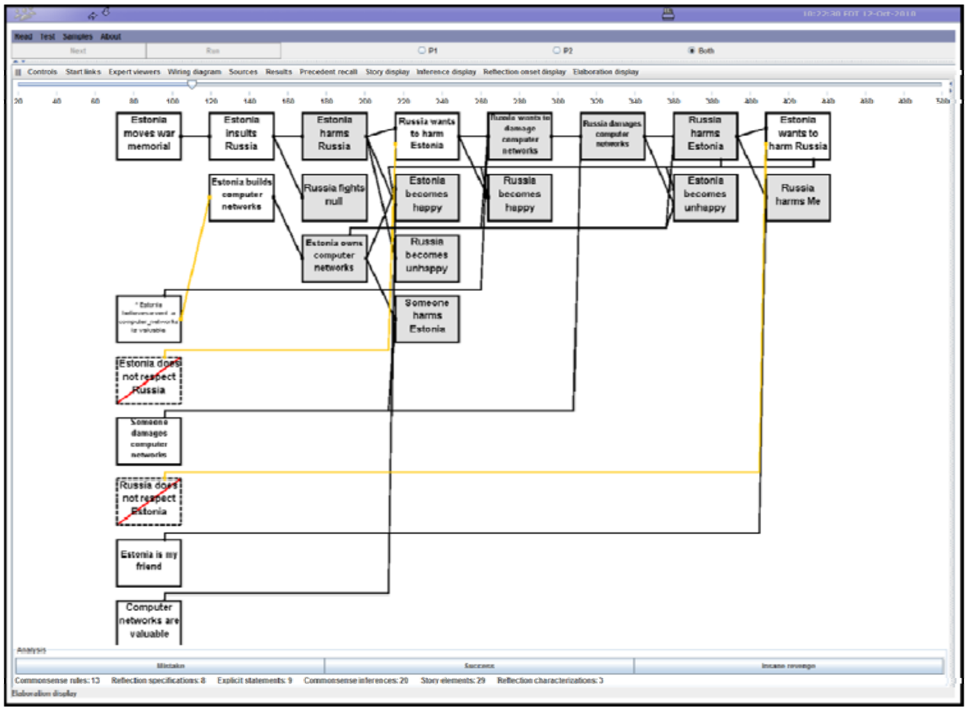


**Figure 4:** An example of an elaboration graph generated from a Story-Workbench-analyzed story. The story in question is Story #3, Russia's cyberattack on Estonia. Boxes colored white indicate information that was explicitly included in the original text of the story. Boxes colored grey indicate information that was inferred by the Commonsense reasoned. Black lines indicate explicit causal connections or explanations, while yellow lines indicate inferred causal connections or explanations.

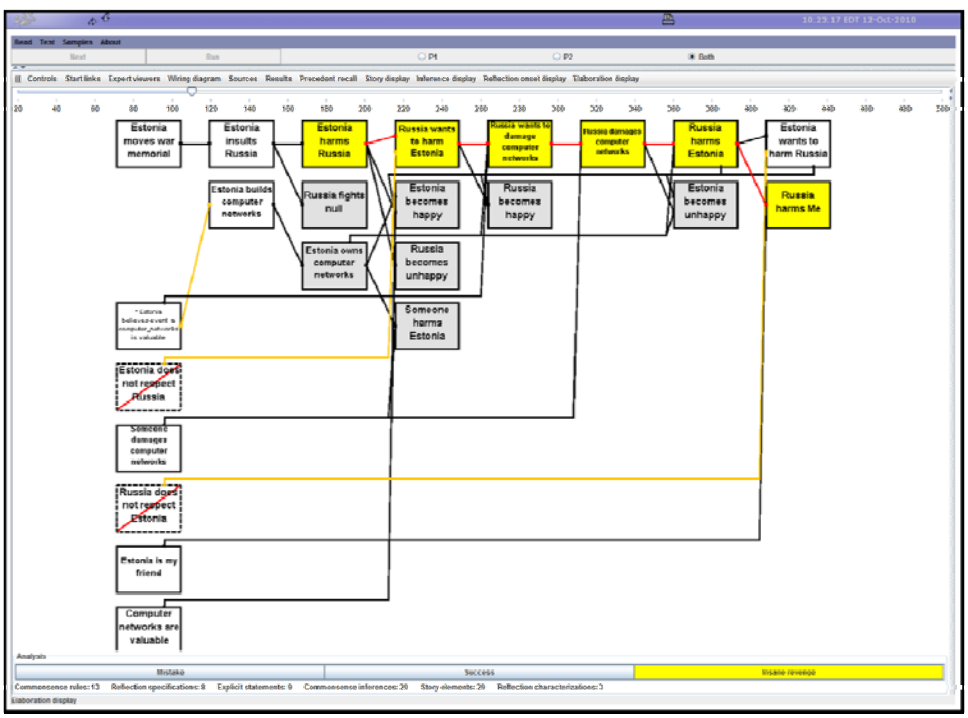


**Figure 5:** Genesis finds an instance of revenge in an elaboration graph. The yellow boxes indicate the portions of the graph implicated in the revenge.

# 4. Further Work

In our proposal we outlined future work involving two more experiments, namely work 10 Shakespearean dramas (experiment #2), and stories from a culture of interest (experiment #3). We began work toward these experiments by using the Story Workbench, and a team of trained annotators, to annotate approximately 20,000 words of Russian folktales. This narrative corpus will serve as a foundation on which to continue our work. In particular, this corpus is the largest, most extensively annotated corpus of narratives yet assembled, and represents a unique contribution to the field.

# 5. Conclusions and Contributions

Information systems have proliferated within the military where "information dominance" has been adopted as a key source of competitive advantage. But as everyone now knows from experience: *access to information does not imply effective use of that information.* As easy as it is to be paralyzed by a lack of information, it is just as easy to be paralyzed by the inability to find the relevant information and put it in context. This is true for policy makers and the intelligence analysts who support them, for military commanders in Command Information Centers (CICs), and for warfighters engaged in unconventional, non-kinetic Stability, Security, Transition, and Reconstruction (SSTR) operations.

In the face of a multi-context, multi-representational, high-volume information stream, consumers need help filtering and interpreting what they see. The challenges of SSTR operations in far-flung cultures have required attention to new and unfamiliar contexts (cultural, social, political) that must be understood on a daily basis by decision makers at all levels. Information gathering in the 21st century generates an overwhelming amount of information of all modalities that must be pruned and interpreted.

Our system, and associated experiment, point the way forward to a class of possible technologies that will assist in interpretation and evaluation of situations relevant to military decision makers. In particular, our novel proof-of-concept demonstration shows that the use of *narrative structuring of information* is a feasible enough approach for further study. In our experiment, we demonstrated the discovery and detection of high-level plot patterns of *revenge* and *pyrrhic victory*. These patterns were discovered by the system without any previous knowledge of what types of patterns to expect. We this important stake in the ground, we can envision systems that would learn all sorts of relevant higher-level patterns from incoming information streams, and then would use these patterns to filter, select, and arrange information for decision makers so as to improve decision quality and turn-around time.

The main concrete technical contributions of this project have been:

1. On a technical, infrastructural level, we connected three novel prototype systems in development at MIT into a single, unified system.
2. We demonstrated that it is feasible to extract, automatically, higher-level plot patterns from sets of stories.
3. We annotated a large corpus of 20,000 words of Russian folktales in 16 representations. This corpus is the largest, most deeply-annotated narrative corpus to date, and will serve as a platform for which much important work can be launched.

# 6. Personnel Involved

- Patrick H. Winston, Ford Professor of Computer Science and Artificial Intelligence
- Mark A. Finlayson, Doctoral Candidate, MIT CSAIL
- Brett van Zuiden, Undergraduate Research Assistant, MIT EECS
- Nidhi Kulkarni, Undergraduate Research Assistant, MIT EECS
- Benjamin Frank, Undergraduate Research Assistant, MIT EECS

# 7. Publications

The work described herein produced three publications so far; more are in the pipeline.


- Finlayson, M.A. (2010) Learning Narrative Morphologies from Annotated Folktales, Proceedings of the 1st International Workshop on Automated Motif Discovery in Cultural Heritage and Scientific Communication Texts (AMICUS) Workshop, Vienna, Austria. pp. 99-102
- Finlayson, M.A. and Kulkarni, N. (2011) Detecting Multi-Word Expressions improves Word Sense Disambiguation. Proceedings of the 2011 Workshop on Multiword Expressions, Portland, Oregon.
- Kulkarni, N. and Finlayson, M.A. (2011) jMWE: A Java Toolkit for Detecting Multi-Word Expressions. Proceedings of the 2011 Workshop on Multiword Expressions, Portland, Oregon.

# 8. References

Angele, J. (2003). Ontonova: Ontology-Based Novel Question Answering; Project Halo Final Report. Karlsruh, Germany, Ontroprise GmbH.

Borchardt, G. C. (1994). Thinking between the Lines: Computers and the Comprehension of Causal Descriptions. Cambridge, MA, MIT Press.

Jackendoff, R. (1983). Semantics and Cognition. Cambridge, MA, MIT Press.

Katz, B., J. J. Lin, et al. (2002). The START Multimedia Information System: Current Technology and Future Directions. International Workshop on Multimedia Information Systems, Tempe, Arizona.

Lehnert, W. G. (1981). "Plot Units and Narrative Summarization." Cognitive Science Volume 5, Issue : 293-331.

Pustejovsky, J., Castano, J., et al. (2003) TimeML: Robust specification of event and temporal expressions in text. In Proceedings of IWCS-5, the Fifth International Workshop on Computational Semantics.

Vaina, L. and R. G. Greenblatt (1979). The Use of Thread Memory in Amnesia Aphasia and Concept Learning. Cambridge, MA, MIT Artificial Intelligence Laboratory Technical Report 195.

# Appendix

## *A. Stories used in the Experiment*

Actual text of the stories used in the experiment. The first, second, fourth, and fifth stories were used to extract the relevant patterns, which were then tested against the third and sixth stories.

### (1) Revenge #1

In early 2010, Google's servers were attacked by Chinese hackers. As such, Google decided to withdraw from China, removing its censored search site and publically criticizing the Chinese policy of censorship. In response, a week later China banned all of Google's search sites.

### (2) Revenge #2

In 1998, Afghan terrorists bombed the U.S.'s embassy in Cairo, killing over 200 people and 12 Americans. Two weeks later, The U.S. retaliated for the bombing with cruise missile attacks on the terrorist's camps in Afghanistan, which were largely unsuccessful. The terrorists claimed that the bombing was a response to America torturing Egyptian terrorists several months earlier.

### (3) Revenge #3 (Test target)

In 2007, Estonia chose to relocate The Bronze Soldier of Talinn, a controversial statue, from the city center to a nearby cemetery. While a seemingly innocuous event, it ended up causing massive political backlash. At the heart of the matter was the controversy around the statue itself: to Russia and ethnic Russian immigrants, it symbolizes the victory of the Soviet Union over Germany in World War II, whereas to many Estonians, it symbolizes Soviet occupation and repression following the war. As such, when the plan to move the statue was announced, many Russians were furious at Estonia, leading to the largest instance of state-sponsored cyber-warfare since Titan Rain. Attacks from Russia (it is unknown whether the attacks were government-sponsored or individuals) caused massive disruption in Estonia, including spamming of Estonian news networks, denial of service attacks against Estonian banks and government organizations, and defacements of the Estonian Reform Party's website. Many Estonians blamed the Russian government for the attacks, but no direct evidence could be found. The incident triggered many military organizations across the world to reconsider the role of network security in the military and national policy.

### (4) Pyrrhic Victory #1

Over the last 10 years, Apple has been trying to increase its market share and popularity, and has succeeded in doing so, with Mac's now comprising two-thirds of the high-end computer market. This increased popularity, however, has also led to increased numbers of malware attacks on Apple's computers, and Apple now recommends using antivirus software.

### (5) Pyrrhic Victory #2

In 2002, as part of its program to maintain strict control over information, China blocked Google entirely. Although they later unblocked it, Google wanted to prevent such an occurrence in the future, and so in 2005 made a compromise with China: Google would filter its search site if China allowed

Google to operate in China. China agreed, but the move caused many to criticize Google for cooperating with China's overbearing censorship policies.

### (6) Pyrrhic Victory #3 (Test target)

In February 2010, Veoh networks, a popular website video company, went bankrupt. The company cited its costly legal battle as the primary cause: even though Veoh won the lawsuit, the distraction and expenses it caused led to Veoh's bankruptcy.

## *B. Genesis Commonsense Knowledge*

The following is the text of the commonsense knowledge used by the Genesis Commonsense Reasoner to infer information necessary for successfully extracting the plot patterns *revenge* and *pyrrhic victory* from our example stories. The common sense knowledge is expressed in English, with comment lines beginning with a double forward slash ('//').

```
// Start Genesis Commonsense Knowledge File

Both perspectives.

Clear story memory.

Clear text.

Start commonsense knowledge.

Henry, George, James, and Mary are persons.
BB is anything.
XX, YY, ZZ, and FF are entities.
CO is a company.
CC is a country.
AA is America.
TT and SS are terrorists.

// can't use operate in CC else the move
// meaning kicks in and we don't get a match
CO prevented CC from blocking CO because CC allows CO to operate CC.

// Representation
// we can't say "originates in" because the "in" does not stay
// within the because block
// XX represents YY because XX originates YY.
If ZZ harms XX and ZZ represents YY then YY harms XX.

If XX owns FF and YY harmed FF, then YY harmed XX.
// getting even by proxy
XX may attack FF because YY harms XX and YY owns FF.
XX may attack FF because YY angers XX and YY owns FF.

// this is a hack to get AA to point to our instance of America
// We assume that the story will have "America's" or "American __"
// somewhere... need to find a way to do this better
XX harms AA because XX harms Americans and AA owns FF.
Estonia owns XX because XX is estonian.
```

```
XX represents Estonia because XX is estonian.
China owns XX because XX is chinese.
XX represents China because XX is chinese.
XX harms Russia because XX angers russians.

// terrorists work together. Ideally this would generalize.
// Again we use the "owns" hack
XX harms TT because XX harms SS and TT owns FF.

// state vs. corporation politics
CO may decide to withdraw from CC because CC harms CO.
If CO withdraws from CC then CO harms CC.
CC may ban XX because CO harms CC and CO owns XX.
CC harms CO because CC bans XX and CO owns XX.

// If XX performs an action and the action causes disruption
// in CC then XX harms CC.
// wanting
If XX wants an action and the action occurs then XX becomes happy.
If XX tries an action then XX wants the action to occur.

// Reasons to kill.
James may kill Henry because James is crazy and James likes Henry.
Henry may want to kill James because Henry is angry at James.

// Friends.
If James harmed George and George is Henry's friend, then James harmed Henry.

// Succession.
If George is king and Henry is George's successor and George becomes dead,
then Henry becomes king.
Mary becomes the queen because George becomes the king and Mary is George's
wife.
James becomes happy because James became the king and James wants to become
the king.
James may murder Henry because James wants to become king and because Henry
is the king.

// Harm.
If XX harms YY, then YY becomes unhappy.
If XX harms YY then XX angers YY.
If YY is furious at XX then XX angers YY.

// Murder killing, and harming
If someone kills you, then you become dead.
James harms Henry because James kills Henry.
James harms Henry because James attacks Henry.
XX harms ZZ because XX attacks ZZ.

//fighting is mutual
Henry fights James because James fights Henry.
James harms Henry because James fights Henry.
James harms Henry because James harasses Henry.
James may attack Henry because Henry harms James.
James may fight Henry because Henry attacks James.
Henry may fight James because Henry is angry at James.
James may kill Henry because James is angry at Henry.
```

```
James may kill Henry because James fights Henry.
XX harms ZZ because XX criticizes ZZ.
XX harms ZZ because XX tortures ZZ.

// helping and happiness
If James helps Henry, then Henry becomes happy.

// Greed
Mary may want to become the queen because she is greedy.

// Persuasion
// If Mary wants an action, then Mary may persuade James to
// commit the action. If Mary persuades James to act, then James acts.

Start commonsense knowledge.

Henry, George, James, and Mary are persons.

First perspective.

James may kill Henry because James is not sane.
James may attack XX because James is not sane.

Second perspective.

Henry may fight James because Henry is angry at James.

Both perspectives.

James may kill himself because James is not sane.
Henry may fight James because Henry is angry at James.
```

# List of Acronyms

| | |
|---|---|
| ASM | Analogical Story Merging |
| NLP | Natural Language Processing |
| TimeML | Time Markup Language |